ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Лекція 19.
Динамічне виділення пам'яті

Дуже часто виникають завдання обробки масивів даних, розмірність яких заздалегідь невідома. У цьому випадку можливе використання одного з двох підходів:

- виділення пам'яті під статичний масив, що містить максимально можливу кількість елементів, проте в цьому випадку пам'ять витрачається нераціонально;
- динамічне виділення пам'яті для зберігання масиву даних.

Стандартні функції динамічного виділення пам'яті

Функції динамічного виділення пам'яті знаходять в оперативній пам'яті безперервну ділянку необхідної довжини і повертають початкову адресу цієї ділянки.

Функції динамічного розподілу пам'яті:

void* malloc(розмір масиву в байтах**);**

void* calloc(число елементів, розмір елемента в байтах**);**

Для використання функцій динамічного виділення пам'яті необхідно підключення бібліотеки **<malloc.h>**

Для визначення розміру використовується функція **int sizeof (тип);**

Пам'ять, яка динамічно виділена з використанням функцій calloc (), malloc (), може бути звільнена з використанням функції **free(покажчик);**

СИНТАКСИС:

(тип*) **malloc(**РозмірМасивуВБайтах);

(int*)malloc(n * sizeof(int));

(тип*)**calloc(**ЧислоЕлементів, РозмірЕлементуВБайтах);

(int*)calloc(n, sizeof(int));

```c
void arrprint(int *a, int);
int main() { srand(time(NULL));
    int arr1[5] = { 1, 7, 13, 42, 5 };
    arrprint(arr1, 5);
    int n, *arr2, *arr3;
    scanf("%d", &n);
    arr2 = (int*)malloc(sizeof(int)*n);
    for (int i = 0; i < n; i++)
        arr2[i]=rand()% 11;
    arrprint(arr2, n);
    arr3 = (int*)calloc(n, sizeof(int));
    for (int i = 0; i < n; i++)
        arr3[i] = rand() % 11+10;
    arrprint(arr3, n);
    free(arr2); free(arr3);
return 0;
}
void arrprint(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%4d ", *(a+i));
    printf("\n");
}
```

Функції calloc і malloc виділяють блоки пам'яті, функція malloc виділяє задане число байт, тоді як calloc виділяє і ініціалізує нулями масив елементів заданого розміру.

```c
void arrprint(int *a, int);
int main() {
    srand(time(NULL));
    int n, *arr3;
    scanf("%d", &n);
    arr3 = (int*)calloc(n, sizeof(int));
    arrprint(arr3, n);
    for (int i = 0; i < n; i++)
        arr3[i] = rand() % 11+10;
    arrprint(arr3, n);
    free(arr3);
return 0;
}
void arrprint(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%4d ", *(a+i));
    printf("\n");
}
```

```
10
   0    0    0    0    0    0    0    0    0    0
  15   15   10   20   15   19   19   14   15   19
```

ПРИКЛАД №1

Динамічні двовимірні масиви

Нехай потрібно розмістити в динамічної пам'яті масив, що містить n рядків і m стовпців. Двовимірний масив буде розташовуватися в оперативній пам'яті у формі стрічки, що складається з елементів рядків. При цьому індекс будь-якого елементу матриці можна отримати за формулою

index = i * m + j

де i - номер поточного рядка; j - номер поточного стовпця.

Наприклад: Знайти індекс виділеного елемента масиву nxm

index = i * m + j

index = 2*4+0=8

7 a[0][0]	**-2** a[0][1]	**0** a[0][2]	**5** a[0][3]
12 a[1][0]	**8** a[1][1]	**17** a[1][2]	**-5** a[1][3]
-2 a[2][0]	**47** a[2][1]	**18** a[2][2]	**6** a[2][3]
1 a[3][0]	**31** a[3][1]	**11** a[3][2]	**3** a[3][3]

7 a[0][0]	**-2** a[0][1]	**0** a[0][2]	**5** a[0][3]	**12** a[1][0]	**8** a[1][1]	**17** a[1][2]	**-5** a[1][3]	**-2** a[2][0]	**47** a[2][1]	**18** a[2][2]	**6** a[2][3]	**1** a[3][0]	**31** a[3][1]	**11** a[3][2]	**3** a[3][3]
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Правильне звернення до елементу масиву з використанням покажчика буде виглядати наступним чином:

***(p + i * m + j),**

де

p - покажчик,

m - кількість стовпців,

i - індекс рядка,

j - індекс стовпця.

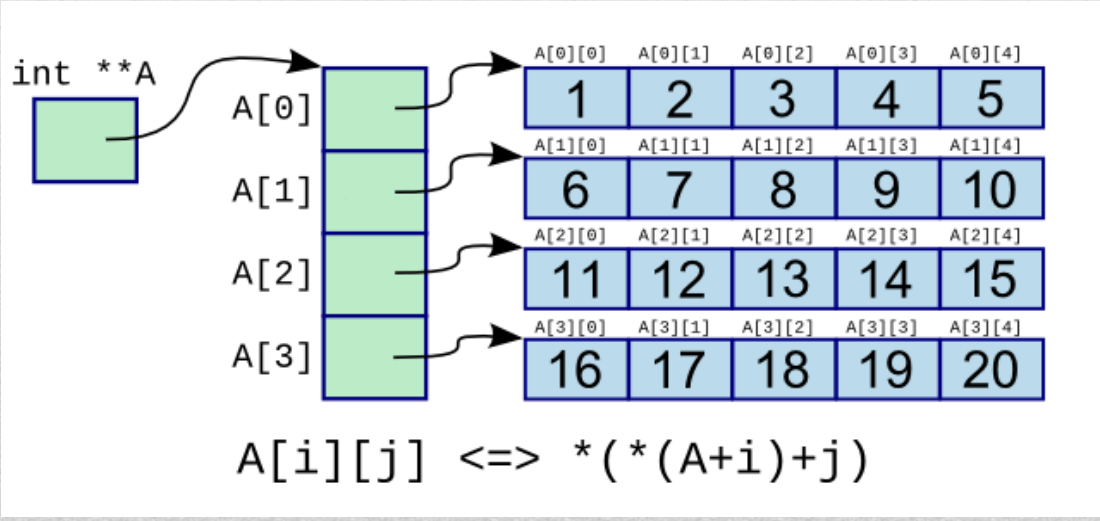
ПРИКЛАД №2

```
int main() {
    system("chcp 1251"); system("cls");
    srand(time(NULL));
    int *a, n, m;
    printf("Введіть кількість рядків: ");
    scanf("%d", &n);
    printf("Введіть кількість стовпців: ");
    scanf("%d", &m);

    a = (int*)malloc(n*m * sizeof(int));
    for (int i = 0; i < n; i++){      // цикл по рядкам
        for (int j = 0; j < m; j++)   // цикл по стовпцям
                *(a + i * m + j) = rand() % 11;
    }

    for (int i = 0; i < n; i++){  // цикл по рядкам
        for (int j = 0; j < m; j++)  // цикл по стовпцям
                printf("%5d ", *(a + i * m + j));
        printf("\n");
    }
    free(a);
return 0;
}
```

```
Введіть кількість рядків: 4
Введіть кількість стовпців: 5
    9     4     0     8     9
   10     9     2     9     9
    5     0     5     7     2
    6     1     5     4    10
```

Можливий також інший спосіб динамічного виділення пам'яті під двовимірний масив - з використанням масиву покажчиків.
Для цього необхідно:

- виділити блок оперативної пам'яті під масив покажчиків;
- виділити блоки оперативної пам'яті під одномірні масиви, що представляють собою рядки матриці;
- записати адреси рядків в масив покажчиків.



```c
int main() {
    system("chcp 1251"); system("cls"); srand(time(NULL));
    int **a, n, m;
    printf("Введіть кількість рядків: "); scanf("%d", &n);
    printf("Введіть кількість стовпців: "); scanf("%d", &m);
    // Виділення пам'яті під покажчики на рядки
    a = (int**)malloc(n * sizeof(int*));
    for (int i = 0; i < n; i++) {     // цикл по рядкам
        a[i] = (int*)malloc(m * sizeof(int)); // Виделення пам’яті під
        зберігання рядків
        for (int j = 0; j < m; j++) {  // цикл по столбцам
                a[i][j] = rand() % 10;
        }
    }

    for (int i = 0; i < n; i++){  // цикл по рядкам
        for (int j = 0; j < m; j++)  // цикл по стовпцям
                printf("%5d ", a[i][j]);
        printf("\n");
    }
    free(a);
return 0;
}
```

realloc

Функция – realloc (re-allocation) дозволяє змінити розмір раніше виділеної пам'яті і отримати в якості аргументів старий покажчик і новий розмір пам'яті

ПРИКЛАД №3

```
void printMas(int *a, int);

int main(){
int *a = NULL, i = 0, elem;
char c;
do {   printf("\na[%d]= ", i); scanf("%d", &elem);
       a = (int*)realloc(a, (i + 1) * sizeof(int));
       a[i] = elem; i++;
    printf("Next (y/n)?");
    c=_getche();
} while (c == 'y');
printMas(a, i);
if (i > 2) i -= 2;
a = (int*)realloc(a, i * sizeof(int)); // зменшення розміру масивуа
printMas(a, i);

return 0;
}
void printMas(int *a, int n) {
printf("\n");
for (int j = 0; j < n; j++)
printf("%d ", j[a]);
}
```

ПРИКЛАД №4

```
system("chcp 1251");
system("cls");
    char *ptr;
    ptr = (char*)realloc(NULL, 20 * sizeof(char));
    strcpy(ptr, "Це перша частина, ");
    printf("%s\n", ptr);

    ptr = (char*) realloc(ptr, 100 * sizeof(char));
    strcat(ptr, "це друга частина");
    printf("%s\n", ptr);
    realloc(ptr, 0);
    //-------------------------------------------------
    ptr = (char*) malloc (20 * sizeof(char));
    strcpy(ptr, "Це третя частина, ");
    printf("%s\n", ptr);
    ptr = (char*)realloc(ptr, 100 * sizeof(char));
    strcat(ptr, "це четверта  частина");
    printf("%s\n", ptr);
    realloc(ptr, 0);
```

```c
typedef double(*arr2d)[5]; //новий тип даних –
                           //покажчик на двовимірний масив

int main() {
    int n;
    scanf("%d", &n);
    arr2d m;
    m = (arr2d)malloc(n * 5 * sizeof(double));
    for (int i = 0; i < n; i++)
        for (int j = 0; j < 5; j++)
                m[i][j] = (rand()%10) / 10.0;
    for (int i = 0; i < n; i++){
        for (int j = 0; j < 5; j++)
                printf("%8.2f", m[i][j]);
        printf("\n");
    }
    free(m);
return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include<time.h>
#include <malloc.h>

int cmp(const void *a, const void *b) {
return *(int*)a - *(int*)b;
}
void arrprint(int *a, int);

int main() {
    srand(time(NULL));
    int n, *a;
    scanf("%d", &n);
    a = (int*)malloc(sizeof(int)*n);
    for (int i = 0; i < n; i++)
        a[i]=rand()% 11;
    arrprint(a, n);   qsort(a, n, sizeof(int), cmp); arrprint(a, n);
return 0;
}

void arrprint(int *a, int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", a[i]);
    printf("\n");
}
```

?